Pattern Recognition Systems – Lab 7

Principal Component Analysis

1. *Objectives*

This laboratory describes the Principal Component Analysis method. It is applied as a means of dimensionality reduction, compression and also visualization. A library capable of providing the eigenvalue decomposition is required.

2. *Theoretical Background*

You are given a set of data points lying in a high dimensional space. Each vector holds the features of a training instance. Our goal is to reduce the dimensionality of the data points while preserving as much information as possible.

We begin with a simple 2D example. We plot the points corresponding to data collected about how different people enjoy certain activities and their skill in the respective domain. Figure 1 shows a cartoon example.

Consider now the two vectors *u_1* and *u_2*. If we project the 2D points onto the vector *u_2* we obtain scalar values with a small spread (standard deviation). If instead, we project it onto *u_1* the spread is much larger. If we had to choose a single vector we would prefer to project onto *u_1* since the points can still be discerned from each other.

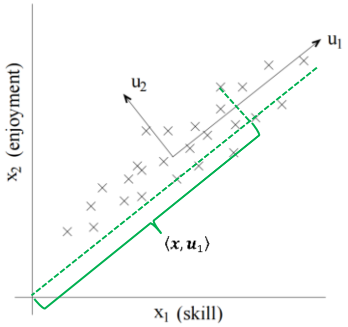

Figure 1. Projection of $\boldsymbol{x}$ along the $\boldsymbol{u}_1$ axis

More formally, each 2D point can be written as:

$$\boldsymbol{x} = \langle \boldsymbol{x}, \boldsymbol{u}_1 \rangle \boldsymbol{u}_1 / \|\boldsymbol{u}_1\| + \langle \boldsymbol{x}, \boldsymbol{u}_2 \rangle \boldsymbol{u}_2 / \|\boldsymbol{u}_2\|$$

Here we have projected $\boldsymbol{x}$ onto each vector and then added the corresponding terms. The dot product $\langle \boldsymbol{x}, \boldsymbol{u}_i \rangle$ gives the magnitude of the projection, it needs to be normalized by the length of the vector $\|\boldsymbol{u}_i\|$ and the two vectors give the directions. This is possible since $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are perpendicular. If we impose that they be also unit vectors then the normalization term disappears. See [4] for a better visualization.

The idea behind reducing the dimensionality of the data is to use only the largest projections. Since the projections onto $\boldsymbol{u}_2$ will be smaller we can approximate $\boldsymbol{x}$ with only the first term:

$$\tilde{\boldsymbol{x}}_1 = \langle \boldsymbol{x}, \boldsymbol{u}_1 \rangle \boldsymbol{u}_1 / \|\boldsymbol{u}_1\|$$

In general, given an orthonormal basis for a *d*-dimensional vector space called ***B*** with basis vectors ***b****ᵢ* we can write any vector as:

$$\boldsymbol{x} = \sum_{i=1}^{d} \langle \boldsymbol{x}, \boldsymbol{b}_i \rangle \boldsymbol{b}_i = \sum_{i=1}^{d} (\boldsymbol{x}^T \boldsymbol{b}_i)\, \boldsymbol{b}_i$$

The question arises of how to determine the basis vectors onto which to perform the projections. Since we are interested in maximizing the preserved variance the covariance matrix could offer useful information. The covariance of two features is defined as:

$$C(x_i, x_j) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu_i)(x_j - \mu_j)$$

where μ_i is the mean for feature *i*. The covariance matrix contains covariance values for all pairs. It can be shown that it can be expressed as a simple matrix product:

$$C = \frac{1}{n-1} (X - \boldsymbol{\mu} \mathbf{1}_{1xn})^T (X - \boldsymbol{\mu} \mathbf{1}_{1xn})$$

where $\boldsymbol{\mu}$ is a vector containing all mean values and $\mathbf{1}_{1xn}$ is a row vector containing only ones. If we extract the mean from the data as a preprocessing step the formula simplifies further:

$$C = \frac{1}{n-1} X^T X$$

The next step is to find the axes along which the covariance is maximal. Eigenvalue decomposition of a matrix offers such information. Intuitively, (almost) any matrix can be viewed as a rotation followed by a stretching along the axes and the inverse rotation. The eigenvalue decomposition returns such a decomposition for the matrix:

$$C = Q \Lambda Q^T = \sum_{i=1}^{d} \lambda_i Q_i Q_i^T$$

where *Q* is a *dxd* rotation matrix (orthonormal) and *Λ* contains elements only on the diagonal representing stretching along each axis. The elements are called eigenvalues and each corresponding column from *Q* is its eigenvector. Since we want to preserve the projections with the largest variance we order the eigenvalues according to magnitude and pick the first *k* corresponding eigenvalues. In this way *C* can be approximated as:

$$\tilde{C}_k = Q_{1:k} \Lambda_{1:k} Q_{1:k}^T = \sum_{i=1}^{k} \lambda_i Q_i Q_i^T$$

where $Q_{1:k}$ is a *dxk* matrix with the first *k* eigenvectors and $\Lambda_{1:k}$ is a *kxk* diagonal matrix with the first *k* eigenvalues. If *k* equals *d* we obtain the original matrix and as we decrease *k* we get increasingly poorer approximations for *C*.

Thus we have found the axes along which the variance of the projections is maximized. Then, for a general vector its approximate with *k* vectors can be evaluated as:

$$\tilde{\boldsymbol{x}}_{\boldsymbol{k}} = \sum_{i=1}^{k} \langle \boldsymbol{x}, Q_i \rangle Q_i = \sum_{i=1}^{k} (\boldsymbol{x}^T Q_i)\, Q_i$$

where Q_i is the *i*th column of the rotation matrix *Q*.

The PCA coefficients can be calculated as:

$$X_{coef} = XQ$$

PCA approximation can be performed on all the input vectors at once (if they are stored as rows in *X*) using the following formulas:

$$\tilde{X}_k = \sum_{i=1}^{k} XQ_iQ_i^t = \sum_{i=1}^{k} X_{coef_i}Q_i^t = XQ_{1:k}Q_{1:k}^T$$

It is important to distinguish the approximation from the coefficients; the approximation is the sum of coefficients multiplied by the principal components.

We will end the theoretical description by giving several examples for application of PCA:

- reducing the dimensionality of features - in some cases large feature vector may prohibit fast prediction;
- visualizing the data - we can inspect only data in 3D and 2D, for higher dimensional data a projection is necessary;
- approximating the data vectors;
- detecting redundant features and linear dependencies between features;
- noise reduction - if the noise term has less variance then the data (high signal-to-noise ratio) PCA eliminates the noise from the input

3. Example results

For pca2d

- First eigenvalue is 8102.21
- Mean absolute difference using only one dimension: 142.8648

For pca3d

- First eigenvalue is 5462.3301
- Mean absolute difference using only one dimension: 116.0689

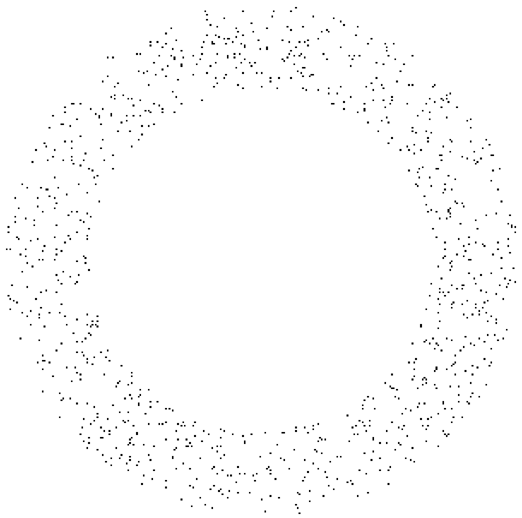

Figure 2. Visualization of 2D points resulting from applying PCA on data from pca2d.txt

4. Implementation details

Declare and allocate an *nxd* matrix with double precision floating point values:

```
Mat X(n,d,CV_64FC1);
```

Calculate the covariance matrix after the means have been subtracted:

```
Mat C = X.t()*X/(n-1);
```

Perform eigenvalue decomposition on the covariance matrix *C*, *Lambda* will contain the eigenvalues and *Q* will contain the eigenvectors along columns.

```
Mat Lambda, Q;
eigen(C, Lambda, Q);
Q = Q.t();
```

Dot product is implemented as normal multiplication. Note that due to 0 indexing the first row is row(0). The dot product between row *i* of *X* and column *i* of *Q* is given by:

```
Mat prod = X.row(i)*Q.col(i);
```

5. Practical Work

1. Open the input file and read the list of data points. The first line contains the number of points *n* and the dimensionality of the data *d*. The following *n* lines each contain a single point with *d* coordinates.
2. Calculate the mean vector and subtract it from the data points.
3. Calculate the covariance matrix as a matrix product.
4. Perform the eigenvalue decomposition on the covariance matrix.
5. Print the eigenvalues.
6. Calculate the PCA coefficients and k^{th} approximate $\tilde{X}_k$ for the input data.
7. Evaluate the mean absolute difference between the original points and their approximation using *k* principal components.
8. Find the minimum and maximum along the columns of the coefficient matrix.
9. For the input data from *pca2d.txt* select the first two columns from X_{coef} and plot the data as black 2D points on a white background. To obtain positive coordinates subtract the minimum values.
10. For input data from *pca3d.txt* select the first three columns from X_{coef} and plot the data as a grayscale image. Use the first two components as x and y coordinates and the third as intensity values. To obtain positive coordinates subtract the minimum values from the first two coordinates. Normalize the third component to the interval 0:255
11. Automatically select the required *k* which retains a given percent of the original variance. For example, find *k* for which the k^{th} approximate retains 99% of the original variance. The percentage of variance retained is given by $\sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{d} \lambda_i$.

6. References

[1] Wikipedia article PCA -
https://en.wikipedia.org/wiki/Principal_component_analysis

[2] Stanford Machine Learning course notes -
http://cs229.stanford.edu/notes/cs229-notes10.pdf

[3] Lindsay Smith - PCA tutorial -
http://faculty.iiit.ac.in/~mkrishna/PrincipalComponents.pdf

[4] PCA in R (animation of projection) -
https://poissonisfish.wordpress.com/2017/01/23/principal-component-analysis-in-r/